Appendix

WINDOWS SCRIPTING HOST

Talk to any UNIX administrator for long enough, and he or she will mention that the Windows operating system lacks a good method of scripting programs. Until recently, this statement was true, but then Microsoft released Windows Scripting Host (WSH).

Before WSH became available, scripted programs were extremely difficult to work with. You could easily create a script via a batch file that displayed information on the screen or executed some internal programs. But if you tried to create one that uses variables and then pass those variables to and from the system, you would notice that the batch file is lacking in some functionality. Although third-party tools are available to improve the ability to create logon scripts, they still lack some network connectivity, such as choosing the server to log into or the domain controller to use for authentication purposes.

The Microsoft WSH engine is a language-independent scripting tool. It runs on all 32-bit Windows platforms and is built into Windows 2000. Best of all, it's free. WSH relies on scripting engines, which provide its power and flexibility. When WSH ships from Microsoft, it includes two scripting engines: Visual Basic Scripting Edition and JavaScript. Microsoft has designed this tool so that third-party developers can also develop their own scripting engines for it, such as Perl, REXX, and Python.

This appendix describes some of the capabilities of WSH. Unfortunately, the actual scripting languages are beyond the scope of this book; therefore, these languages will not be covered in detail.

WSH comes in two flavors: a text-based, DOS-like version (Cscript.exe) and a Windows-based version (Wscript.exe). These versions are covered in the next sections.

COMMAND-BASED SCRIPTING

The command line-based version of WSH, called Cscript.exe, is located in the %systemroot%\system32 directory. The application is executed using the following syntax:

```
CScript scriptname.extension [option...] [arguments...]
```

The Cscript switches are as follows:

- **Scriptname.extension** This switch consists of the name of the script file and its extension—for example, script.js (for JavaScript) or script.vbs (for Visual Basic). This script is a text file that can be edited with any basic text editor. Other extensions exist for third-party scripting languages that might be installed in the system.
- **Option** Options allow you to customize how your script will be executed. You can enable or disable any of the options, as noted in Table A-1. Be aware that any option must be preceded by two forward slashes (//). You may sometimes find these options referred to as *host parameters*.
- **Arguments** Some options have the ability to receive information from the operating system and send it through to the script. These are called arguments.

For example, to execute a JavaScript script that shuts down a server (it is named shutdown.js), with it being given a maximum of 30 seconds to execute with no input from any users, you would issue the following command:

```
Script shutdown.js //B //T:30
```

The "//B" option sets the script into batch mode and suppresses all errors and messages. The "//T:30" option passes an argument of 30 seconds to the //T option (which will time the script out after 30 seconds).

Table A-1 Options for Cscript.exe

Option	Description
//B	Sets the WSH into batch mode. All script errors and prompts are suppressed and do not display.
//D	Enables Active Debugging mode.
//E:engine	Allows you to set the engine to be used for executing the script.
//H:Cscript	Changes the default script host to Cscript.exe.
//H:Wscript	Changes the default script host to Wscript.exe (the default).
//I	Puts WSH into Interactive mode (the default and the opposite of //B). All script errors and messages are displayed.
//Job:xxxx	Executes a WSH job.
//Logo	Displays the logo (the default). A banner will be shown at the execution time of the script.
//Nologo	Prevents the logo from displaying. No banner will be shown at the execution time of the script.

Table A-1 Options for Cscript.exe (continued)

Option	Description
//S	Saves the current command-line options for this user.
//T:nn	Time out in seconds. Sets a maximum amount of time that a script is permitted to run before WSH terminates it.
//U	Redirects the output from the script.
//X	Executes the script in the debugging mode.

WINDOWS-BASED SCRIPTING

Windows-based scripting is slightly more intuitive. The command line is pretty much the same:

```
Wscript scriptname.extension [option...] [arguments...]
```

Table A-2 lists all of the available options. Notice that, except for lacking the //X option, all of the options exist in both types of WSH scripts.

Table A-2 Options for Wscript.exe

Option	Description
//B	Sets the WSH into batch mode. All script errors and prompts are suppressed and do not display.
//D	Enables Active Debugging mode.
//E:engine	Allows you to set the engine to be used for executing the script.
//H:Cscript	Changes the default script host to Cscript.exe.
//H:Wscript	Changes the default script host to Wscript.exe (the default).
//I	Puts WSH into Interactive mode (the default and the opposite of //B). All script errors and messages are displayed.
//Job:xxxx	Executes a WSH job.
//Logo	Displays the logo (the default). A banner will be shown at the execution time of the script.
//Nologo	Prevents the logo from displaying. No banner will be shown at the execution time of the script.
//S	Saves the current command-line options for this user.
//T:nn	Time out in seconds. Sets a maximum amount of time that a script is permitted to run before WSH terminates it.
//U	Redirects the output from the script.

Manually Registering Script Engines

Sometimes, you may find that you need to manually configure a new scripting engine or reconfigure an existing one. This section deals with how your would register such an engine, assuming that:

- The scripting engine is called **NewScriptEngine**.
- The file extension used for the script engine is **.nsf**.
- The scripting identifier is **NewScriptFile**.

Table A-3 lists all of the Registry entries that must be added to register the new script.

Table A-3 Manually registering a new script engine

Registry Key	Key Type	Key Value
.nsf	REG_SZ	NewScriptFile
NewScriptFile	REG_SZ	NewScriptEngine Script File
NewScriptFile\ScriptEngine	REG_SZ	NewScriptEngine
NewScriptFile\Shell Open\	REG_SZ	&Open
NewScriptFile\Shell Open\ Command	REG_EXPAND_SZ	%systemroot%\system32\ wscript "%1" %*
NewScriptFile\Shell Open2	REG_EXPAND_SZ	Open &with command console
NewScriptFile\Shell Open2\ Command	REG_SZ	%systemroot%\system32\ wscript "%1" %*
NewScriptFile\ShellEx\ PropertySheetHandlers\ WSHProps	REG_SZ	{60254CA5-953B-11CF-8C96-00AA00B8708C}

Sample Script

The following is a simple script using Visual Basic Script to display a message and then count to 10.

```
Dim Counter

Wscript.Echo "Welcome to my first Windows Scripting Host
program"

Wscript.Echo "Here is a script that will count from 1 to 10"

For Counter = 1 to 10

Wscript.Echo Counter

Next
```